Discrete Applied Mathematics 14 (1986) 269–281
North-Holland

UNIMODULAR FUNCTIONS

Pierre HANSEN
RUTCOR, Rutgers University, Hill Center, New Brunswick, NJ 08903, USA

Bruno SIMEONE*
Department of Statistics, University of Rome, Italy

Received 19 October 1976
Revised November 1984 and 8 November 1985

We introduce some classes of pseudo-boolean functions (the so-called 'unimodular', 'completely unimodular' and 'unate' ones), whose maximization over the binary n-cube is reducible to a maximal flow problem.

All such classes of functions are generalizations of classes previously investigated by Rhys and Balinski.

It is shown that in the quadratic case the above three classes coincide, and that they also coincide with the class of those pseudo-boolean functions f such that a certain signed graph G_f associated with f is balanced.

The latter characterization leads to a polynomial recognition algorithm.

When G_f is a (signed) tree, a linear-time maximization algorithm is available.

1. Introduction

Let B^n be the *binary n-cube*, i.e. the n-th cartesian power of the set $B=\{0,1\}$. *A pseudo-boolean function* is any mapping f from B^n into the set R of reals. It is well-known [9] that any such function has a unique representation as a multilinear polynomial in n variables:

$$f(x)=\sum_{T\in\mathcal{T}} a_T \prod_{i\in T} x_i \tag{1.1}$$

where $\mathcal{T}$ is a collection of subsets of $N=\{1,\ldots,n\}$.

The problem of maximizing an arbitrary pseudo-boolean function f over B^n is known to be NP-complete [6]. However, there are special cases in which efficient solution algorithms do exist. One such case occurs when the coefficients of all the non-linear terms of $f(x)$ are non-negative, i.e. when $a_T\geq 0$ for all $T\in\mathcal{T}$ such that $|T|\geq 2$. A pseudo-boolean function f having this property will be called *almost-positive* (of course, f is allowed to have linear terms with negative coefficients).

*Current address: RUTCOR, Rutgers University, New Brunswick, NJ 08903, USA.

0166-218X/86/$3.50 © 1986, Elsevier Science Publishers B.V. (North-Holland)

Hammer and Rudeanu [8], Rhys [20], Balinski [1], Picard and Ratliff [19] have established the mutual reducibility (in polynomial time) between the maximization of an almost-positive function and the maximal network flow problem. Since the latter problem can be solved in $O(v^3)$ time [4], [16], [17], where v is the number of nodes in the network, it follows that the maximum of an almost-positive function can be found in polynomial time.

It is of interest to identify, if possible, larger classes of pseudo-boolean functions, whose maximization is reducible to a maximal flow problem.

One cannot hope to go too far beyond the class of almost-positive functions, even in the quadratic case. Indeed, it has been shown in [10] that the problem of maximizing the quadratic function x^TQx is NP-complete even when Q is upper triangular and has at most one negative element per row.

A natural generalization of almost-positive functions is obtained by considering those pseudo-boolean functions which can be transformed into almost-positive ones by 'switching' some variables.

Such functions will be called unate.

A different generalization of almost-positive functions is suggested by Rhys' observation that the maximization of any such function is reducible to a linear program with totally unimodular matrix. Generalizing Rhys' construction, with any given pseudo-boolean function f we associate a 'canonical' family L_f of linear programs. If L_f contains a member with totally unimodular matrix, f is called unimodular. If *all* members of L_f are such that their matrix is totally unimodular, then f is said to be completely unimodular. It is not hard to see that the maximization of a unimodular function is reducible to a maximal flow (in fact, to a minimum cut) problem.

In Section 2 we analyze the mutual relationships between the classes of unimodular, completely unimodular and unate functions. In particular, the class of unimodular functions is seen to include the other two classes properly.

The main result of this paper (Section 3) is that in the quadratic case the three above classes coincide. Furthermore, we prove that a quadratic function f is unimodular if and only if a certain signed graph G_f associated with f is balanced. This characterization leads to a linear-time recognition algorithm for quadratic unimodular functions.

Some of these results were anticipated in [11] and in [21].

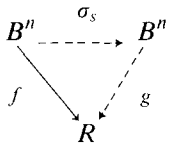
The special case when G_f is a (signed) tree is of interest. In Section 4 this case is investigated and a linear-time maximization algorithm is described.

2. Unimodular functions

Let S denote a subset of $N=\{1,\ldots,n\}$.

The *switch* on S is the mapping $\sigma_S: B^n \to B^n$ which maps the vector x into the vector y defined by $y_i=x_i$, $\forall i \notin S$ and $y_i=\bar{x}_i$, $\forall i \in S$.

Let us call a pseudo-boolean function f *unate* if there exist an almost-positive function g and a subset S of $N=\{1,\ldots,n\}$ such that the diagram



commutes; in other words if f can be represented as an almost-positive function of n variables, the i-th variable being either x_i or its *complement* $\bar{x}_i = 1 - x_i$.

Example 1. The function $f(x) = -x_1x_2x_3 + x_1x_2 - x_1x_3 - x_2x_3 + x_1 + x_2$ is unate, since $f(x) = x_1x_2\bar{x}_3 + x_1\bar{x}_3 + x_2\bar{x}_3$, which is an almost-positive function in the variables $x_1, x_2, \bar{x}_3$.

It is clear that, if f is unate, the problem $\max_{x\in B^n} f(x)$ can be formulated as a maximal flow one.

A second possible way of generalizing almost-positive functions is suggested by Rhys' proof of the fact that the maximization of any such function is reducible to a maximal flow problem.

Suppose that the almost-positive function f is given by

$$f(x) = \sum_{T\in\mathcal{T}} a_T \prod_{i\in T} x_i + \sum_{i=1}^{n} b_i x_i \tag{2.1}$$

where $|T| \geq 2$, $a_T > 0$ for all $T \in \mathcal{T}$, and without loss of generality $b_i < 0$ (if $b_j > 0$ for some index, then there is a maximizer x^* of f such that $x_j^* = 1$).

Rhys observed [20] that the maximum value of f in B^n is equal to the optimal value of the linear program

$$\begin{aligned} \max \quad & \sum_{T\in\mathcal{T}} a_T y_T + \sum_{i=1}^{n} b_i x_i, \\ \text{s.t.} \quad & y_T \leq x_i, \qquad T\in\mathcal{T}, i\in T, \\ & 0 \leq x_i \leq 1, \qquad i = 1,\ldots,n, \\ & 0 \leq y_T, \qquad T\in\mathcal{T}. \end{aligned} \tag{2.2}$$

This result is an immediate consequence of the following two remarks:

(1) For every fixed *binary* $x\in B^n$, the optimal value of (2.2) - in which x is regarded as a parameter - is equal to $f(x)$, since the coefficients a_T are positive and

$$\min_{i\in T} x_i = \prod_{i\in T} x_i \quad \text{for all } T\in\mathcal{T}.$$

(2) The linear program (2.2) has an optimal integral solution, since its matrix is totally unimodular.

Actually, (2.2) is the dual of a maximal flow problem. We notice that Rhys' proof relies on the fact that all the coefficients a_T are positive.

In order to extend these considerations to more general pseudo-boolean functions, we need the further concept of pseudo-disjunctive normal form.

Hammer and Rosenberg have observed [7] that a pseudo-boolean function f can always be represented as

$$f(x) = c + q(x, \bar{x}), \tag{2.3}$$

where c is a constant and q is a *posiform*, i.e. a polynomial with positive coefficients in the original variables $x_1, \ldots, x_n$ and in their complements $\bar{x}_1, \ldots, \bar{x}_n$. The representation (2.3) is not unique. Any pair (c, q) satisfying (2.3) for all $x \in B^n$ is called in [21] a *pseudo-disjunctive normal form* (PDNF) of f.

Now, let f be a pseudo-boolean function, and let us consider a PDNF (c, q) of f. We assume that the posiform q is *primitive*, in the sense that no two distinct terms of q involve the same set of variables, no matter whether complemented or not (for example the presence of a term $x_1x_2x_3$ forbids the presence of $\bar{x}_1x_2x_3$, $x_1\bar{x}_2x_3, \ldots, \bar{x}_1\bar{x}_2\bar{x}_3$). If q is not primitive, we can always obtain from (c, q), via simple transformations, a new PDNF (c', q') of f, such that q' is primitive. Since the coefficients of q are positive, we may regard q as an almost-positive function of the $2n$ variables $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$; thus we can associate with q a linear program L_q in the way suggested by Rhys.

More precisely, let

$$q(x, \bar{x}) = \sum_{(S,T)\in\mathscr{P}} a_{ST} \prod_{i\in S} x_i \prod_{j\in T} \bar{x}_j + \sum_{i\in I} b_i x_i + \sum_{j\in J} c_j \bar{x}_j \tag{2.4}$$

where $\mathscr{P}$ is a collection of subsets of $N \times N$ such that

$$S \cap T = \emptyset,\ |S \cup T| \geq 2 \text{ and } a_{ST} > 0 \quad \text{for all } (S, T) \in \mathscr{P};$$

$$I \cap J = \emptyset,\ I \cup J \subseteq N,\ b_i > 0 \quad \text{for all } i \in I \quad \text{and} \quad c_j > 0 \quad \text{for all } j \in J.$$

Then the linear program associated with q is

$$\begin{aligned}
\max \quad & \sum_{(S,T)\in\mathscr{P}} a_{ST} y_{ST} + \sum_{i\in I} b_i x_i + \sum_{j\in J} c_j \bar{x}_j, \\
\text{s.t.} \quad & y_{ST} \leq x_i, && (S, T) \in \mathscr{P},\ i \in S, \\
& y_{ST} \leq \bar{x}_j, && (S, T) \in \mathscr{P},\ j \in T, \\
& y_{ST} \geq 0, && (S, T) \in \mathscr{P}, \\
& 0 \leq x_i \leq 1,\ 0 \leq \bar{x}_i \leq 1, && i = 1, \ldots, n.
\end{aligned}$$

Replacing $\bar{x}_i$ by $1 - x_i$ $(i = 1, \ldots, n)$, we obtain a linear program in the variables y_{ST}, x_i:

$$
\begin{aligned}
(L_q) \qquad & \sum_{j\in J} c_j + \max \sum_{(S,T)\in \mathscr{P}} a_{ST} y_{ST} + \sum_{i\in I} b_i x_i - \sum_{j\in J} c_j x_j, \\
\text{s.t.} \quad & y_{ST} - x_i \le 0, \quad (S,T)\in \mathscr{P}, i\in S, \\
& y_{ST} + x_j \le 1, \quad (S,T)\in \mathscr{P}, j\in T, \\
& y_{ST} \ge 0, \quad (S,T)\in \mathscr{P}, \\
& 0 \le x_i \le 1, \quad i = 1,\ldots,n
\end{aligned}
\tag{2.5}
$$

or, in matrix form,

$$
\begin{aligned}
(L_q) \qquad & c_0 + \max d \begin{bmatrix} y \\ \cdots \\ x \end{bmatrix}, \\
\text{s.t.} \quad & M \begin{bmatrix} y \\ \cdots \\ x \end{bmatrix} \le p, \\
& y \ge 0, \\
& 0 \le x \le e,
\end{aligned}
\tag{2.6}
$$

where e is the vector $(1,\ldots,1)$.

All elements of the matrix M are 0, 1, -1 and each row of M has exactly two non-zero elements. However, in general M is not totally unimodular. A pseudo-boolean function f is said to be *unimodular* whenever it admits a primitive PDNF (c, q) such that the matrix M of the linear program L_q is totally unimodular. By a well-known theorem of Heller–Tompkins–Gale [14], a necessary and sufficient condition for M to be totally unimodular is that the set of columns of M can be partitioned into two subsets Q_1, Q_2 (one of which might be empty) such that:

(i) If the non-zero elements of a row have the same sign, one of the corresponding columns belongs to Q_1 and the other one to Q_2.

(ii) If the two non-zero elements of a row have opposite signs, the corresponding columns belong to a same subset Q_i.

Example 2. The function

$$f(x) = -x_1x_2x_3 - x_1x_2x_4 - x_1x_3x_4 - x_2x_3x_4 + 2x_1x_4 + 2x_2x_3$$

admits the PDNF $(0, q)$, where $q = \bar{x}_1x_2x_3 + x_1\bar{x}_2x_4 + x_1\bar{x}_3x_4 + x_2x_3\bar{x}_4$.

The associated linear program L_q is given by (2.6), with

$$M = \begin{array}{c} \begin{matrix} y_{123} & y_{124} & y_{134} & y_{234} & x_1 & x_2 & x_3 & x_4 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}, \quad p = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$d = [\, 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \,], \quad c_0 = [\, 0 \,].$$

A direct application of the Theorem of Heller-Tompkins-Gale shows that M is totally unimodular: actually, one may take Q_1 as the set of the columns corresponding to $y_{123}, y_{234}, x_2, x_3$, and Q_2 as the set of the remaining columns. Hence f is unimodular.

When the matrix M is totally unimodular, by switching all the variables corresponding to columns in Q_2 one obtains a linear program in which all the constraints have the from $u \leq v$. Picard has shown [18] that linear programs of this kind are reducible to minimum cut problems.

When f has the stronger property that, for *every* primitive PDNF (c, q) of f, the matrix M of the corresponding linear program L_q is totally unimodular, then f is said to be *completely unimodular.*

Let us now turn our attention to the mutual relationships between the classes of unate, unimodular and completely unimodular functions.

Proposition 1. *Every unate function is unimodular.*

Proof. Since f is unate, there exists a posiform q of the form (2.4) such that $f(x) = q(x, \bar{x})$ for all $x \in B^n$, and having the property that there is a bipartition $\{J_1, J_2\}$ of N such that $S \subseteq J_1$ and $T \subseteq J_2$ for all $(S, T) \in \mathscr{P}$. It follows that all the constraints of the linear program L_q are either of the form $y_{ST} - x_i \leq 0$, where $i \in J_1$, or of the form $y_{ST} + x_j \leq 1$, where $j \in J_2$.

Partition the set of columns of M into two subsets Q_1, Q_2 as follows.

(1) Q_1 is the set of those columns corresponding to variables y_{ST} or to variables x_i with $i \in J_1$.

(2) Q_2 is the set of those columns corresponding to variables x_j with $j \in J_2$. It is easy to see that the bipartition $\{Q_1, Q_2\}$ satisfies the conditions of the above men-

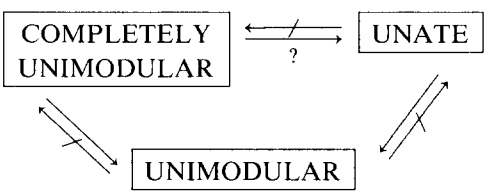
tioned Heller-Tompkins-Gale Theorem. Hence M is totally unimodular and f is unimodular. □

However, a unate function does not have to be completely unimodular. The function f given in Example 1 is unate. Yet f is not completely unimodular, since f admits the PDNF $(-2, q)$, where $q(x, \bar{x}) = \bar{x}_1 x_2 x_3 + x_1 x_2 + x_1 \bar{x}_3 + 2x_2 x_3 + x_2 + 2\bar{x}_3$, and it is easily checked that the matrix M of the corresponding linear program L_q is not totally unimodular.

In general, a unimodular function does not have to be unate: for instance, the unimodular function f given in Example 2 is not unate.

Open question. Are there completely unimodular functions that are not unate?

The mutual relationships between unimodular, completely unimodular and unate functions are summarized in the following diagram. (An arrow from Class A to Class B means that Class A is contained in Class B.)



Finally, we notice that, since the right-hand side of the linear program L_q is a binary vector, this program would retain the integrality property in the more general case of M being a perfect matrix [2].

3. The quadratic case

The main result of this section states that, in the case of quadratic functions, the classes of unimodular, completely unimodular and unate functions coincide: a graph-theoretic characterization of such functions is also given, in terms of a signed graph which represents the sign pattern of their coefficients.

We recall that a pseudo-boolean function f is *supermodular* if for all $x, y \in B^n$

$$f(x \vee y) + f(x \wedge y) \geq f(x) + f(y).$$

It is shown in [5] that a quadratic pseudo-boolean function is almost-positive if and only if it is supermodular.

Moreover, Billionnet and Minoux [3] have recently shown that the reducibility of the maximization of a supermodular function to a maximal flow problem still holds in the cubic case.

Before proving the main result, we need a preliminary lemma:

Lemma 2. *Let f be a quadratic pseudo-boolean function and (c, q) any quadratic primitive pseudo-disjunctive normal form of f. Then f has a monomial cxy, $c>0$, if and only if q has a monomial cxy or $c\bar{x}\bar{y}$; and f has a monomial $-cxy$, $c>0$, if and only if q has a monomial $cx\bar{y}$ or $c\bar{x}y$.*

Proof. If we replace in q all complemented variables $\bar{x}_i$ by $1-x_i$, we obtain a quadratic polynomial in the n variables $x_1, \ldots, x_n$. The thesis then follows from the uniqueness of representation of f as a polynomial in $x_1, \ldots, x_n$. □

Now let

$$f(x)=x^{\mathrm{T}}Qx=\sum_{h,k=1}^{n} q_{hk}x_hx_k$$

be a quadratic pseudo-boolean function. Let us associate with f the signed graph G_f whose vertices are the n variables $x_1, \ldots, x_n$, and where two vertices x_i and x_j are linked by a *positive* edge if $q_{ij}>0$ and by a *negative* edge if $q_{ij}<0$.

If γ is a bipartition, into positive and negative edges, of the edges of a graph G, G is defined to be *balanced* with respect to γ if it has no cycle with an odd number of negative edges. A well-known theorem of Harary [13] states that G is balanced if and only if there exists a bipartition of its vertices into positive and negative ones such that:

(a) Vertices of the same sign are connected only by positive edges.

(b) Vertices of different signs are connected only by negative edges.

Theorem 3. *For a quadratic pseudo-boolean function f, the following propositions are equivalent:*

(1) *f is unimodular.*

(2) *The signed graph associated with f is balanced.*

(3) *f is unate.*

(4) *f is completely unimodular.*

The nature of the proof is graph-theoretical: the basic tool is Theorem 4 below.

If $G=(V, E)$ is a graph, the *bigraph* of G is the graph $B(G)$ whose vertices are the elements of $V \cup E$ and whose edges are all (v, e) such that $v \in V$, $e \in E$ and v is incident with e. $B(G)$ is a bipartite graph and each of its vertices belonging to E has degree 2. We shall say that an edge-bipartition β of $B(G)$ *agrees* with the edge-bipartition γ of G if:

(i) For any vertex b of $B(G)$ which corresponds to a negative edge of G, the two edges of $B(G)$ incident with b have different signs in β.

(ii) For any vertex w of $B(G)$ which corresponds to a positive edge of G, the two edges of $B(G)$ incident with w have the same sign in β.

Theorem 4. *The graph G is balanced with respect to γ if and only if its bigraph $B(G)$ is balanced with respect to any β which agrees with γ.*

Proof. Any cycle $C = v_1 v_2 \cdots v_n v_1$ in G corresponds to the cycle $C' = v_1 e_1 v_2 e_2 \cdots v_n e_n v_1$ where $e_1 = v_1 v_2$, $e_2 = v_2 v_3, \ldots, e_n = v_n v_1$. For any β which agrees with γ, the parity of the number of edges in C which are negative in γ equals the parity of the number of edges in C' which are negative in β. Hence the theorem follows. □

Corollary 5. *If $B(G)$ is balanced with respect to some β which agrees with a given bipartition γ of the edges of G, then $B(G)$ is balanced with respect to any β' which agrees with γ.*

We can now prove Theorem 3.

Proof of Theorem 3. Let us give a graph-theoretic interpretation of the conditions (1), (2), (3). To this aim, we introduce the graph $G = (V, E)$ whose vertex set is $V = \{v_1, \ldots, v_n\}$ and in which two vertices v_h and v_k are connected by an edge whenever $q_{hk} \neq 0$. Define γ to be the bipartition of the edges of G obtained by assigning to the edge (h, k) the sign $-$ or $+$ according to whether q_{hk} is negative or positive. Let now (c, q) be any quadratic primitive PDNF of $f(x)$. In view of Lemma 2, the linear program (2.5) corresponding to q has the following structure. Its variables are y_{ij}, $\forall\, (v_i, v_j) \in E$ and x_j, $\forall v_j \in V$. For each monomial $q_{ij} x_i x_j$, the linear program has two rows: namely, when $q_{ij} > 0$, the two rows are

$$\begin{array}{lcl} & y_{ij} \cdots x_i \cdots x_j & \qquad y_{ij} \cdots x_i \cdots x_j \\ \text{either} & 1 \cdots -1 \cdots & \text{or} \quad 1 \cdots 1 \cdots \\ & 1 \cdots \cdots -1 & \qquad 1 \cdots \cdots \quad 1, \end{array}$$

depending on whether the corresponding term in q is $x_i x_j$ or $\bar{x}_i \bar{x}_j$, and

$$\begin{array}{l} y_{ij} \cdots x_i \cdots x_j \\ 1 \cdots -1 \cdots \\ 1 \cdots \cdots \quad 1 \end{array}$$

when $q_{ij} < 0$ and q has a monomial $q_{ij} x_i \bar{x}_j$.

The columns of M correspond precisely to the vertices of the bigraph $B(G)$, while the rows correspond to the edges of $B(G)$. Let us define a bipartition β of the edges of $B(G)$ as follows: an edge is negative or positive according to whether the two non-zero entries of the corresponding row in M have the same sign or different signs, respectively. It is clear that β agrees with the bipartition γ in G previously defined.

Combining the theorems of Heller-Tompkins-Gale and of Harary, it is easily seen that the matrix M of (2.5) is totally unimodular if and only if $B(G)$ is balanced with respect to β. Thus, in the statement of Theorem 2, the condition (1): "f is unimodular" amounts to the existence of some β which agrees with γ and such that $B(G)$ is balanced with respect to β.

On the other hand, condition (2) is equivalent to the balance property of G with respect to γ.

Condition (3) is equivalent to: "There exists a partition of V into two classes S and $V-S$ such that vertices belonging to different classes are connected only by negative edges, and vertices belonging to the same class are connected only by positive edges". (Then S is the set required by the definition of unateness.) From the above considerations it follows that (1) and (2) are equivalent by Theorem 3, and conditions (2) and (3) are equivalent by Harary's theorem.

Finally, we observe that, if f is unimodular then, in view of Corollary 5, for every quadratic primitive PDNF (c, q) of f, the matrix M of the corresponding linear program L_q is totally unimodular: in other words, this property is *invariant* with respect to the quadratic primitive PDFN's of f. Hence (1) and (4) are equivalent. □

We notice that condition (3) of Theorem 2 yields an effective test for deciding whether or not a given quadratic pseudo-boolean function is unimodular. In fact, the balance of a signed graph with m edges can be checked in $O(m)$ time by using a simple implementation of Algorithm 1 in [12].

4. Tree-functions

In the present section we introduce a special class of unimodular quadratic functions of n binary variables that can be maximized in $O(n)$ time.

A function f is called a *tree-function* if G_f is a (signed) tree.

Therefore, a tree-function has the form

$$f(x)=\sum_{(i,j)\in E(T)} a_{ij}x_ix_j+\sum_{i\in V(T)} b_ix_i \tag{4.1}$$

where T is a tree, $a: E(T)\to R-\{0\}$ and $b: V(T)\to R-\{0\}$.

The problem of maximizing over B^n the function f given by (4.1) will be denoted by $P(T, a, b)$. We regard the tree T as rooted, the root being an arbitrary vertex of T. *The predecessor* of vertex i will be denoted by $p(i)$. By condition (2) of Theorem 2, every tree-function is unate.

The following simple recursive algorithm identifies a set $S\subseteq V(T)$ such that the switch on S transforms f into a supermodular function.

Algorithm

Step 1. Assign the label $L(r)=1$ to the root r

Step 2. Until there are no unlabelled vertices, do the following:

Let i be any unlabelled vertex such that its predecessor $p(i)$ is labelled; if $(i, p(i))$ is a positive edge, assign to i the label $L(i)=L(p(i))$; else assign to i the label $L(i)=-L(p(i))$.

By construction, vertices with the same label are linked by positive edges, while vertices with different labels are linked by negative edges.

Hence, the switch on the set $S \equiv \{i : L(i) = -1\}$ transforms f into a supermodular function. Thus we can assume, from now on, that all the coefficients a_{ij} in (4.1) are positive.

Proposition 6. *Let i be a leaf of T. There is an optimal solution x^* of $P(T, a, b)$ such that*

(1) *If $b_i \geq 0$, then $x_i^* = 1$.*

(2) *If $b_i < 0$ and $a_{ip(i)} + b_i \leq 0$, then $x_i^* = 0$.*

(3) *If $b_i < 0$ and $a_{ip(i)} + b_i > 0$, then $x_i^* = x_{p(i)}^*$.*

Proof. In case (1) one has, for all $x \in B^n$,

$$f(x_1, \ldots, \overset{i}{1}, \ldots, x_n) - f(x_1, \ldots, \overset{i}{0}, \ldots, x_n) = a_{ip(i)} x_{p(i)} + b_i \geq 0.$$

In case (2) one has, for all $x \in B^n$,

$$f(x_1, \ldots, \overset{i}{0}, \ldots, x_n) - f(x_1, \ldots, \overset{i}{1}, \ldots, x_n) = -a_{ip(i)} x_{p(i)} - b_i \geq 0.$$

In case (3) one has, for all $x \in B^n$,

$$f(x_1, \ldots, \overset{i}{x_{p(i)}}, \ldots, x_n) - f(x_1, \ldots, \overset{i}{1 - x_{p(i)}}, \ldots, x_n) =$$

$$= (a_{ip(i)} + b_i) x_{p(i)} - b_i (1 - x_{p(i)}) \geq 0.$$

Proposition 7. *Let i be a leaf of T, $T^1 = T - i$, a^1 the restriction of a to $E(T^1)$, and let b^1 be defined as follows:*

In case (1) *of Proposition* 6,

$$b_j^1 = \begin{cases} b_j, & j \in V(T^1),\ j \neq p(i), \\ b_{p(i)} + a_{ip(i)}, & j = p(i). \end{cases}$$

In case (2), *b^1 is the restriction of b to $V(T^1)$.*

In case (3),

$$b_j^1 = \begin{cases} b_j, & j \in V(T^1),\ j \neq p(i), \\ b_{p(i)} + b_i + a_{ip(i)}, & j = p(i). \end{cases}$$

Finally, let x^* be an optimal solution of $P(T^1, a^1, b^1)$.

Then the vector $\tilde{x}$ defined by

$$\tilde{x}_j = \begin{cases} x_j^*, & j \in V(T^1), \\ 1, & j = i, \text{ case (1) of Proposition 6}, \\ 0, & j = i, \text{ case (2) of Proposition 6}, \\ x_{p(i)}^*, & j = i, \text{ case (3) of Proposition 6}. \end{cases}$$

is an optimal solution of $P(T, a, b)$.

Proof. The proof follows immediately from Proposition 6 and from the identities

$$f(x_1, \ldots, \overset{i}{1}, \ldots, x_n) = \sum_{(k,j)\in E(T^1)} a_{kj}x_k x_j + \sum_{j\in V(T^1)} b_j x_j + a_{ip(i)}x_{p(i)} + b_{p(i)}.$$

$$f(x_1, \ldots, \overset{i}{0}, \ldots, x_n) = \sum_{(k,j)\in E(T^1)} a_{kj}x_k x_j + \sum_{j\in V(T^1)} b_j x_j.$$

$$f(x_1, \ldots, \overset{i}{x_{p(i)}}, \ldots, x_n) = \sum_{(k,j)\in E(T^1)} a_{kj}x_k x_j + \sum_{j\in V(T^1)} b_j x_j + a_{ip(i)}x_{p(i)} + b_i x_{p(i)}.$$

Proposition 7 allows for the recursive solution of $P(T, a, b)$ by reducing it to a sequence of problems $P(T', a', b')$, $P(T'', a'', b'')$, ... on smaller and smaller trees $T', T'', \ldots$, until the trivial tree with a single vertex is obtained. The resulting algorithm can be implemented so as to run in $O(n)$ time. Here are the details (we follow the terminology of [22]). A *breadth-first order* in $V(T)$ is a linear order $<$ in $V(T)$ such that

(a) If the depth of i is smaller than the depth of j, then $i<j$.

(b) If i and j are brothers and $i<k<j$, then k is a brother of i and j.

Algorithm

Let T be given by its adjacency list.

Step 1 (Initialization). For $i=1, \ldots, n$, compute the predecessor $p(i)$ of i.

For $k=1, \ldots, n$ let $q(k)$ be the k-th vertex in a breadth-first order in $V(T)$. Set $w(i) \leftarrow b_i$, $x_i \leftarrow -1$ for all i.

Comment. $w(\cdot)$ is the vector of vertex weights in the current tree, x will eventually contain the solution of $P(T, a, b)$.

Step 2 (Shrink). For $k=n, n-1, \ldots, 2$ do the following:

Let $i=q(k)$;

Comment. i is a leaf of the current tree.

If $b_i \geq 0$ then $x_i \leftarrow 1$, $w(p(i)) \leftarrow w(p(i)) + a_{ip(i)}$;

If $b_i<0$ and $a_{ip(i)} + b_i \leq 0$, then $x_i \leftarrow 0$;

If $b_i<0$ and $a_{ip(i)} + b_i > 0$, then $w(p(i)) \leftarrow w(p(i)) + w(i) + a_{ip(i)}$ and mark the edge $(i, p(i))$;

Repeat.

At the end, let $r=q(1)$;

If $w(r) \geq 0$, then $x_r \leftarrow 1$, else $x_r \leftarrow 0$.

Step 3 (Blow-up). For $k=2, \ldots, n-1, n$ do the following:

Let $i=q(k)$;

If the edge $(i, p(i))$ is marked, then $x_i \leftarrow x_{p(i)}$.

Comment. $x_{p(i)}$ is necessarily 0 or 1.

Repeat.

End

Clearly, each step requires $O(n)$ elementary operations,
Hence the overall complexity of the algorithm is $O(n)$.

References

[1] M.L. Balinski, On a selection problem, Manag. Sci. 17 (1970) 230–231.

[2] C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1973).

[3] A. Billionet and M. Minoux, Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions, Discrete Appl. Math. 12 (1985) 1–11.

[4] B.V. Cherkasky, Algorithm of construction of maximal flow in networks with complexity of $O(|V|^2 \cdot |E|^{1/2})$ operations, Math. Methods of Solution of Econ. Problems 7(1977) 117–125.

[5] M.L. Fisher, G.L. Nemhauser and L.A. Wolsey, An analysis of approximations for maximizing submodular set functions, I, Math. Programming 14 (1978) 265–294.

[6] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, 1978).

[7] P.L. Hammer and I.G. Rosenberg, Linear decomposition of a positive group-boolean function, in: L. Collatz and W. Wetterling, eds., Numerische Methoden bei Optimierung, Vol. II (Birkhauser, Basel, 1974) 51–62.

[8] P.L. Hammer and S. Rudeanu, On solving the transportation problem with Egerváry method – II (Rumenian), Studi si Cercetari Matem. 14 (1963) 59–67.

[9] P.L. Hammer and S. Rudeanu, Boolean Methods in Operations Research and related areas (Springer, Heidelberg, 1968).

[10] P.L. Hammer and B. Simeone, Quasimonotone boolean functions and bistellar graphs, Annals Discrete Math. 8 (1980) 107–119.

[11] P. Hansen, Minimization d'une fonction quadratique en variables zéro-un par l'algorithme de Ford et Fulkerson, presented at the Colloque Structures Economiques et Econometrie, Lyon (1973).

[12] P. Hansen, Labelling algorithms for balance in signed graphs, in: J.C. Bermond et al. eds., Problemes Combinatoires et Theorie des Graphes, Colloques Internationaux du CNRS No. 260 (CNRS, Paris, 1978).

[13] F. Harary, On the notion of balance of a signed graph, Michig. Math. J. 2(1953) 143–146.

[14] I. Heller and C.B. Tompkins, On unimodular matrices, Pacif. J. Math. 12(1962) 1321–1327 (with an appendix by D. Gale).

[15] A.J. Hoffman and J.B. Kruskal, Integral boundary points of convex polyhedra, in: H. Kuhn and A.W. Tucker, eds., Linear Inequalities and Related Systems (Princeton Univ. Press, Princeton, 1956) 223–246.

[16] A.V. Karzanov, Determining the maximal flow in a network by the method of preflows, Soviet Math. Doklady 15 (1974) 434–437.

[17] V.M. Malhotra, M.P. Kumar and S.N. Maheshwari, An $O(|V|^3)$ algorithm for finding maximum flows in networks, Inform. Process. Lett. 7(1978) 277–278.

[18] J.C. Picard, Maximal closure of a graph and applications to combinatorial problems, Manag. Science 22(1976) 1268–1272.

[19] J.C. Picard and H. Ratliff, Minimum cuts and related problems, Networks 5(1975) 357–370.

[20] J. Rhys, A selection problem of shared fixed costs and network flows, Manag. Sci. 17(1970) 200–207.

[21] B. Simeone, Quadratic 0–1 programming, boolean functions and graphs, Doctoral dissertation, Univ. of Waterloo, 1979.

[22] T.A. Standish, Data Structures Techniques (Addison-Wesley, Reading, MA 1980).